DBMS used for non relational databases:

There are several types of NoSQL databases, each suited for different use cases. Here are some commonly used NoSQL database management systems (DBMS) along with their types:

Document Stores:

MongoDB: MongoDB is a popular document-oriented NoSQL database that stores data in BSON format (Binary JSON). It is known for its flexibility and scalability.

Key-Value Stores:

Redis: Redis is an in-memory key-value store. It's often used for caching and real-time analytics due to its high-speed read and write operations.

Column-Family Stores:

Apache Cassandra: Cassandra is a distributed NoSQL database designed for scalability and high availability. It's commonly used for time-series data and big data applications.

HBase: HBase is another distributed, scalable, and consistent NoSQL database that is often used with Hadoop for handling large datasets.

Graph Databases:

Neo4j: Neo4j is a popular graph database management system that specializes in storing and querying graph data. It's used for applications involving complex relationships, such as social networks and recommendation engines.

Object Stores:

Amazon S3: While not a traditional NoSQL database, Amazon S3 (Simple Storage Service) is an object store that allows you to store and retrieve objects, making it suitable for various unstructured data storage needs, such as images, videos, and backups.

Time-Series Databases:

InfluxDB: InfluxDB is designed for handling time-series data and is commonly used in monitoring and IoT applications.

Search Engines:

Elasticsearch: Elasticsearch is a distributed search and analytics engine that is commonly used for full-text search and log data analysis.

Multimodel Databases:

ArangoDB: ArangoDB is a multi-model NoSQL database that supports document, key-value, and graph data models in a single database engine.

The return of ‘execute(): ’True/False

The execute method returns false because it's typically used for SQL statements that don't return results (e.g., INSERT, UPDATE, DELETE, CREATE, etc.). If you were to execute a query that returns a result set (e.g., SELECT), it would return true.

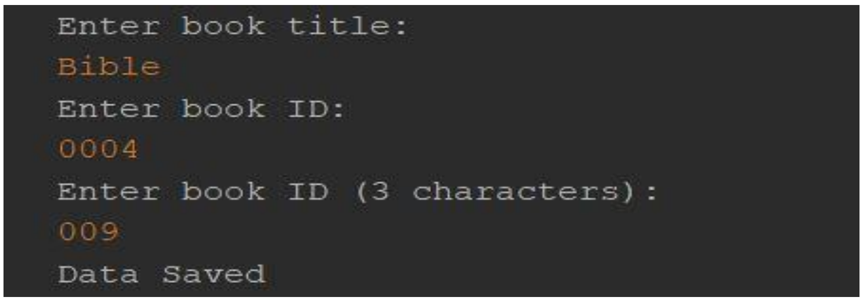
Difference between parameter and argument

A parameter is a variable or placeholder declared in the function or method signature. It represents a value that the function expects to receive when it is called while an argument is an actual value or expression that is passed to a function or method when it is called. It corresponds to the parameters declared in the function's signature.

_ Parameters are defined in the function or method declaration and act as input placeholders for the values that will be passed to the function / Arguments are provided when invoking a function or method, and they supply the actual data that the function will work with

_ Parameters define what kind of data the function will accept and use during its execution. They act as variables within the function, allowing it to work with the provided values / Arguments supply specific values to the function's parameters, allowing the function to perform operations using those values.

Validate user's input on the book_id (3 characters)



```
Enter book title:
Bible
Enter book ID:
0004
Enter book ID (3 characters):
009
Data Saved
```

I used the infinite loop (While) to repeat the statement until the user respects the conditions. Inside the loop we compare the length of the user input and compare it to 3 which is the length of the book_id column in the database. Below is the code which validate the user input:

```
while (true) {
            System.out.println("Enter book ID (3 characters): ");
            bookID = sc.next();
            if (bookID.length() == 3) {
                break; // Exit the loop if the input is valid
            } else {
                System.out.println("Book ID must have exactly 3 characters. Please try again.");
            }
        }
```